

# An All-Around Near-Optimal Solution for the Classic Bin Packing Problem

Shahin Kamali

Alejandro López-Ortiz \*

## Abstract

In this paper we present the first algorithm with optimal average-case and close-to-best known worst-case performance for the classic on-line problem of bin packing. It has long been observed that known bin packing algorithms with optimal average-case performance were not optimal in the worst-case sense. In particular First Fit and Best Fit had optimal average-case ratio of 1 but a worst-case competitive ratio of 1.7. The wasted space of First Fit and Best Fit for a uniform random sequence of length  $n$  is expected to be  $\Theta(n^{2/3})$  and  $\Theta(\sqrt{n} \log^{3/4} n)$ , respectively. The competitive ratio can be improved to 1.691 using the Harmonic algorithm; further variations of this algorithm can push down the competitive ratio to 1.588. However, Harmonic and its variations have poor performance on average; in particular, Harmonic has average-case ratio of around 1.27. In this paper, first we introduce a simple algorithm which we term Harmonic Match. This algorithm performs as well as Best Fit on average, i.e., it has an average-case ratio of 1 and expected wasted space of  $\Theta(\sqrt{n} \log^{3/4} n)$ . Moreover, the competitive ratio of the algorithm is as good as Harmonic, i.e., it converges to 1.691 which is an improvement over 1.7 of Best Fit and First Fit. We also introduce a different algorithm, termed as Refined Harmonic Match, which achieves an improved competitive ratio of 1.636 while maintaining the good average-case performance of Harmonic Match and Best Fit. Finally, our extensive experimental evaluation of the studied bin packing algorithms shows that our proposed algorithms have comparable average-case performance with Best Fit and First Fit, and this holds also for sequences that follow distributions other than the uniform distribution.

---

\*School of Computer Science, University of Waterloo, Canada. Email: {s3kamali, alopez-o}@uwaterloo.ca

# 1 Introduction

An instance of the classical online bin packing problem is defined by a sequence of *items* which are revealed in an online manner. Each item has a size in the range  $(0, 1]$ . The goal is to pack these items into a minimum number of bins which have a uniform capacity of 1. A natural algorithm for the problem is Next Fit (NF) which keeps one *open* bin at each time. If a given item does not fit into the bin, the algorithm *closes* the bin (i.e., it does not refer to it in future) and opens a new bin. In contrast to NF, First Fit (FF) does not close any bin: It maintains the bins in the order they are opened and places a given item in the first bin which has enough space for it. In case such a bin does not exist, it opens a new bin for the item. Best Fit (BF) performs similarly to FF, except that it maintains the bins in decreasing order of their *levels*; the level of a bin is the total size of items placed in the bin. Another approach is to divide items into a constant number of classes based on their sizes and pack items of the same class apart from other classes. An example is the Harmonic (HA) algorithm which has a parameter  $K$  and defines  $K$  intervals  $(1/2, 1]$ ,  $(1/3, 1/2]$ ,  $\dots$ ,  $(1/(K-1), 1/K]$ , and  $(0, 1/K]$ ; items which belong to the same interval are separately treated using the Next Fit strategy.

Online bin packing algorithms are usually compared through their respective average-case performance and worst-case performance. Under average-case analysis, it is assumed that item sizes follow a fixed distribution that is typically a uniform distribution. With this assumption, one can define the *average performance ratio* as the ratio between the expected cost of an online algorithm for a randomly selected sequence compared to the cost of OPT. Here OPT is an optimal offline algorithm with unbounded computational power. It is known that NF has an average performance ratio of  $1.3$  [8] for sequences generated uniformly at random. FF and BF are optimum in this sense and have an average ratio of 1 [3]. To further compare algorithms with average ratio of 1, one can consider the *expected waste* which is the expected amount of wasted space for serving a sequence of length  $n$ . More precisely, the wasted space of an algorithm for serving a sequence  $\sigma$  is the difference between the cost of the algorithm and the total size of items in  $\sigma$ . FF and BF have expected waste of sizes  $\Theta(n^{2/3})$  and  $\Theta(\sqrt{n} \log^{3/4} n)$ , respectively [29, 9, 23]. It is also known that all online algorithms have expected waste of size  $\Omega(\sqrt{n} \lg^{1/2} n)$  [29]. These results show that BF is almost the best online algorithm with respect to average performance.

There are other algorithms which perform almost as well as BF on average. These algorithms are based on matching a ‘large’ item with a ‘small’ item to place them in the same bin. Throughout the paper we call an item *large* if it is larger than  $1/2$  and *small* otherwise. Among the matching-based algorithms are Interval First Fit (IFF) [13] and Online Match (OM) [10]. IFF has a parameter  $K$  and divides the unit interval into  $K$  intervals of equal length, namely  $I_t = (\frac{t-1}{K}, \frac{t}{K}]$  for  $t = 1, 2, \dots, K$ . Here,  $K$  is an odd integer and we have  $K = 2j + 1$ . The algorithm defines  $j + 1$  classes so that intervals  $I_c$  and  $I_{K-c}$  form class  $c$  ( $1 \leq c \leq j$ ) and interval  $I_K$  forms class  $j + 1$ . Items in each class are packed separately from other classes. The items in class  $c$  ( $2 \leq c \leq j + 1$ ) are treated using FF strategy, while the items in the first class are treated using an Almost FF strategy. Almost FF is similar to FF except that it closes a bin when it includes a small and a large item; further, a large item is never placed in a bin which includes more than one small items, and a bin with  $k$  small items is declared as being closed. The average ratio of IFF is 1; precisely, it has an expected waste of  $\Theta(n^{2/3})$ . Algorithm OM has also a parameter  $K$  and declares two items as being *companions* if their sum is in the range  $[1 - \frac{1}{K}, 1]$ . To place a large item, OM opens a new bin. To place a small item  $x$ , the algorithm checks whether there is an open bin  $\beta$  with a large companion of  $x$ ; in case there is, OM places  $x$  in  $\beta$  and closes  $\beta$ . Otherwise, it packs  $x$  using a NF strategy in a separate list of bins. The average ratio of OM converges to 1 for large values of  $K$  [10].

Although the matching-based algorithms have acceptable average performance, they do not perform well in the worst-case. In particular, IFF has an unbounded competitive ratio [13], and the competitive ratio of OM is 2 [10]. Among other matching algorithms we might mention Matching Best Fit (MBF) which performs similarly to BF except that it closes a bin as soon as it receives the first small item. The average

ratio of MBF is as good as BF while it has unbounded competitive ratio [29]. There is another algorithm which has expected waste of size  $\Theta(\sqrt{n} \lg^{1/2} n)$  [30] which matches the lower bound of [29]. This algorithm also has a non-constant competitive ratio [7].

The competitive ratio<sup>1</sup> reflects the worst-case performance of online algorithms. More formally, it is the asymptotically maximum ratio between the cost of an online algorithm and that of OPT for serving the same sequence. It is known that NF has a competitive ratio of 2 while FF and BF have the same ratio of 1.7 [20]. The competitive ratio of HA converges to 1.691 for sufficiently large values of  $K$  [22]. To be more precise, it approaches  $T_\infty = \sum_{i=1}^{\infty} \frac{1}{t_i - 1}$ , where  $t_1 = 2$  and  $t_{i+1} = t_i(t_i - 1) + 1, i > 1$ <sup>2</sup>. There are online algorithms which have even better competitive ratios. These include Modified First Fit (MFF) with ratio 1.666 [33], Modified Harmonic with ratio around 1.635 [22], and Harmonic++ with ratio 1.588 [28]. These algorithms are members of a general framework of Super Harmonic algorithms [28]. Similar to HA, Super Harmonic algorithms classify items by their sizes and pack items of the same class together. However, to handle the bad sequences of HA, a fraction of opened bins include items from different classes. These bins are opened with items of small sizes in the hopes of subsequently adding items of larger sizes. At the time of opening such a bin, it is pre-determined how many items from each class should be placed in the bin. As the algorithm runs, the reserved spot for each class is occupied by an item of that class. It is guaranteed that the reserved spot is enough for any member of the class. This implies that the expected total size of items in the bin is strictly less than 1 by a positive value. Consequently, the expected waste of the algorithm is linear to the number of opened bins. Hence, for a sequence of length  $n$ , these algorithms have an expected waste of  $\Omega(n)$ . Since the expected wasted space of OPT is  $o(n)$ , the average performance ratio of Super Harmonic algorithms is strictly larger than 1. In particular, Refined Harmonic and Modified Harmonic have average performance ratios around 1.28 and 1.18, respectively [32, 25].

Table 1 shows the existing results for major bin packing algorithms. As pointed out by Coffman et al. in [7], ‘*All algorithms that do better than First Fit in the worst-case seem to do much worse in the average case.*’ In this paper, however, we show that this is not a necessary condition and give an algorithm whose average-case ratio, competitive ratio, and expected wasted space are all at or near the top of each class. This also addresses a conjecture in [32] stated as ‘*Harmonic- $K$  is better than First Fit in the worst-case performance, and First Fit is better than Harmonic- $K$  in the average-case performance. Maybe there exists an on-line algorithm with the advantages of both First Fit and Harmonic- $K$ .*’

## Contribution

We introduce an algorithm called Harmonic Match (HM) and show this algorithm is better than BF and FF in the worst case, while it performs as well as BF and FF on average. In particular, we show the competitive ratio of HM is as good as HA, i.e., it approaches  $T_\infty \approx 1.691$  for sufficiently large values of  $K$ . For sequences generated uniformly at random, the average performance ratio of HM is 1, which is as good as BF and FF. The expected waste of HM is  $\Theta(\sqrt{n} \log^{3/4} n)$  which is as good as BF and better than FF. The algorithm is easy to implement and has the same running time as BF.

HM can be seen as a general way to improve the performance of the Super Harmonic class of algorithms in general and Harmonic algorithm in particular. We illustrate this for the simplest member of this family, namely Refined Harmonic algorithm. To do so, we introduce a new algorithm, called Refined Harmonic Match (RHM), and show that the competitive ratio of the algorithm is at most equal to 1.636 of Refined Harmonic, while its average-case ratio is 1 which is as good as BF and HM. The expected waste of RHM is equal to that of BF. Consequently, the algorithm achieves the desired average-case performance of BF and also the worst-case performance of Refined Harmonic.

<sup>1</sup>By competitive ratio, we mean *asymptotic* competitive ratio where the number of opened bins by an optimal offline algorithm is arbitrarily large. For the results related to the *absolute* competitive ratio, we refer the reader to [7, 12].

<sup>2</sup>Some notations are borrowed from [7].

Algorithm	Average Ratio	Expected waste	Competitive Ratio
Next Fit (NF)	$1.\bar{3}$ [8]	$\Omega(n)$	2
Best Fit (BF)	1 [3]	$\Theta(\sqrt{n} \log^{3/4} n)$ [29, 23]	1.7 [20]
First Fit (FF)	1 [23]	$\Theta(n^{2/3})$ [29, 9].	1.7 [20]
Harmonic (HA)	1.2899 [22]	$\Omega(n)$	$\rightarrow T_\infty \approx 1.691$ [22]
Refined First Fit (RFF)	$> 1$	$\Omega(n)$	1.6 [33]
Refined Harmonic (RH)	1.2824 [32]	$\Omega(n)$	1.636[22, 32]
Modified Harmonic (MH)	1.189 [25]	$\Omega(n)$	1.615[26]
Harmonic++	$> 1$	$\Omega(n)$	1.588 [28]
<b>Harmonic Match (RH)</b>	<b>1</b>	<b><math>\Theta(\sqrt{n} \log^{3/4} n)</math></b>	<b><math>\rightarrow T_\infty \approx 1.691</math></b>
<b>Refined Harmonic Match (RHM)</b>	<b>1</b>	<b><math>\Theta(\sqrt{n} \log^{3/4} n)</math></b>	<b><math>&lt; 1.636</math></b>

Table 1: Average performance ratio, expected waste (under continuous uniform distribution), and competitive ratios for different bin packing algorithms. Results in bold are our contributions.

Similar to HA, HM and RHM are based on classifying items based on their sizes and treating items of each class (almost) separately. To boost the average-case performance, these algorithms *match* large items with proportionally smaller items through assigning them to the same classes. Careful definition of classes results in the same average-case performance of MBF. To some extent, our competitive analyses of HM and RHM are similar to those of HA and Refined HA, respectively. Similarly, the average-case analyses of the algorithms are closely related to the analysis of MBF algorithm and uses similar techniques.

To evaluate the average-case performance of the introduced algorithms in real-world scenarios, we tested them on sequences that follow *discrete* uniform distribution as well as other distributions. We compared HM and RHM with the existing algorithms and observed that they have comparable performance with BF and FF. At the same time, these algorithms had a considerable advantage over other members of the Harmonic family of algorithms. We conclude that HM and RHM have better average-case performance than existing algorithms which outperform BF and FF in the worst-case scenarios.

## 2 Harmonic Match Algorithm

Recall that Harmonic (HA) algorithm which has a parameter  $K$  and defines  $K$  classes  $(1/2, 1]$ ,  $(1/3, 1/2]$ ,  $\dots$ ,  $(1/(K-1), 1/K]$ , and  $(0, 1/K]$ ; items in the same class are separately treated using the Next Fit strategy. Similarly to Harmonic algorithm, Harmonic Match has a parameter  $K$  and divides items into  $K$  classes based on their sizes. We use  $\text{HM}_K$  to refer to Harmonic Match with parameter  $K$ . The algorithm defines  $K$  pairs of intervals as follows. The  $i$ th pair ( $1 \leq i \leq K-1$ ) contains intervals  $(\frac{1}{i+2}, \frac{1}{i+1}]$  and  $(\frac{i}{i+1}, \frac{i+1}{i+2}]$ . The  $K$ th pair includes intervals  $(0, \frac{1}{K+1}]$  and  $(\frac{K}{K+1}, 1]$ . An item  $x$  belongs to class  $i$  if the size of  $x$  lies in any of the two intervals associated with the  $i$ th pair (see Figure 1). Intuitively, the items which are ‘very large’ or ‘very small’ belong to the  $K$ th class, and as the item sizes become more moderate, they belong to classes with smaller indices.

When compared to the intervals of Harmonic algorithms, one can see the first interval of the  $i$ th pair in the Harmonic Match algorithm  $\text{HM}_K$  is the same as the  $(i+1)$ th interval of Harmonic algorithm  $\text{HA}_{K+1}$  ( $1 \leq i \leq K$ ). Namely, the intervals are the same in both algorithms except that the interval  $(\frac{1}{2}, 1]$  of HA is further divided into  $K+1$  more intervals. In other words,  $\text{HM}_K$  is similar to  $\text{HA}_{K+1}$ , except that it tries to match large items with proportionally smaller item. The pairs of intervals which define a class in HM have the same length, e.g., in the first pair, both intervals have length  $\frac{1}{6}$ . This property is essential for having good

$$\begin{array}{rcl}
i=1 & \frac{1}{3} < x \leq \frac{1}{2} & \longleftrightarrow \frac{1}{2} < x \leq \frac{2}{3} \\
i=2 & \frac{1}{4} < x \leq \frac{1}{3} & \longleftrightarrow \frac{2}{3} < x \leq \frac{3}{4} \\
i=3 & \frac{1}{5} < x \leq \frac{1}{4} & \longleftrightarrow \frac{3}{4} < x \leq \frac{4}{5} \\
& \vdots & \\
i=k-1 & \frac{1}{k+1} < x \leq \frac{1}{k} & \longleftrightarrow \frac{k-1}{k} < x \leq \frac{k}{k+1} \\
i=k & x \leq \frac{1}{k+1} & \longleftrightarrow x > \frac{k}{k+1}
\end{array}$$

Figure 1: The classes defined by HM. The algorithm matches items from intervals indicated by arrows.

average-case performance.

The packing maintained by HM include two types of bins: the *mature* bins which are almost full and *normal* bins might become mature by receiving more items. For placing an item  $x$ , HM detects the class that  $x$  belongs to and applies the following strategy to place  $x$ . If  $x$  is large item (recall that by large we mean larger than  $\frac{1}{2}$ ), HM opens a new bin for  $x$  and declares it as a normal bin. If  $x$  is small, the algorithm applies BF strategy to place  $x$  in a mature bin. If there is no mature bin with enough space, the BF strategy is applied again to place  $x$  in a normal bin which contains the largest ‘companion’ of  $x$ . A companion of  $x$  is a large item of the same class which fits with  $x$  in the same bin. In case the BF strategy succeeds to place  $x$  in a bin (i.e., there is a normal bin with a companion of  $x$ ) the selected bin is declared as being mature. Otherwise (when there is no companion for  $x$ ), the algorithm applies NF strategy to place  $x$  in a single normal bin maintained for that class; such a bin only includes small items of the class. If the bin maintained by the NF strategy does not have enough space, it is declared as a mature bin and a new NF-bin is opened for  $x$ . Note that HM, as defined above, is simple to implement and its time complexity is as good as BF.

HM treats items of the same class in a similar way that Online Match (OM) algorithm does, except that there is no restriction on the sum of the sizes of two companion items. Recall that OM has a parameter which defines a lower bound for the sum of two items in a bin. To facilitate our analysis in the following sections, we define algorithm Relaxed Online Match (ROM) as a subroutine of HM as follows. To place a large item, ROM opens a new bin. To place a small item  $x$ , it applies the BF strategy to place  $x$  in an open bin with a single large item and closes the bin. If such a bin does not exists, ROM places  $x$  using NF strategy (and opens a new bin if necessary). Using ROM, we can describe Harmonic Match algorithm in the following way. To place a small item,  $HM_K$  tries to place it in a mature bin using BF strategy. Large items and the small items which do not fit in mature bins are treated using ROM strategy along with other items of their classes (which did not fit in mature bins). The bins which are closed by ROM strategy are declared as mature bins.

## 2.1 Worst-Case Analysis

For the worst-case analysis of HM, we observe that the Harmonic algorithm is *monotone* in the sense that removing an item does not increase its cost:

**Lemma 1.** *Removing an item does not increase the costs for the Harmonic algorithm.*

*Proof.* Recall that  $HA_K$  defines a class for each item and applies the NF strategy to place each item together with items of the same class. So, the cost of the algorithm for serving a sequence  $\sigma$  is  $NF(\sigma_1) + NF(\sigma_2) +$

$\dots + \text{NF}(\sigma_K)$ , where  $\sigma_i$  is the sequence of items which belong to class  $i$ . Assume an item  $x$  is removed from  $\sigma$  and let  $j$  denote the class that  $x$  belongs to ( $1 \leq j \leq K$ ). The cost of  $\text{HA}_K$  for serving the reduced sequence (in which  $x$  is removed) will be the same except that  $\text{NF}(\sigma_j)$  is replaced by  $\text{NF}(\sigma'_j)$ , where  $\sigma'_j$  is a copy of  $\sigma_j$  in which  $x$  is missing. Since  $\text{NF}$  is monotone [24], we have  $\text{NF}(\sigma'_j) \leq \text{NF}(\sigma_j)$ . Consequently, the cost of  $\text{HA}$  cannot increase after removing  $x$ .  $\square$

We use the above lemma to show that the cost of  $\text{HM}_K$  for serving any sequence  $\sigma$  is no larger than that of  $\text{HA}_{K+1}$ . Consequently, the competitive ratio of  $\text{HM}_K$  is no larger than that of  $\text{HA}_{K+1}$ .

**Theorem 1.** *The cost of  $\text{HM}_K$  to serve any sequence  $\sigma$  is no larger than that of  $\text{HA}_{K+1}$ .*

*Proof.* Consider the final packing of  $\text{HM}$  for serving  $\sigma$ . Colour a small item *red* if it is packed with a large item in the same bin; colour all other small items *white*. Consider the sequence  $\sigma'$  which is the same as  $\sigma$  except that the red items are removed. We claim  $\text{HM}_K(\sigma) = \text{HA}_{K+1}(\sigma')$ . Let  $\sigma_i$  denote the sequence of items which belong to class  $i$  of  $\text{HM}_K$  ( $1 \leq i \leq K$ ). The cost of  $\text{HM}_K$  for serving  $\sigma_i$  is  $l_i + \text{NF}(W_i)$ , where  $l_i$  is the number of large items  $\sigma_i$  and  $W_i$  is the sequence formed by white items in  $\sigma_i$ . Let  $\sigma'_i$  be a subsequence of  $\sigma_i$  in which red items are removed (hence it is also a subsequence of  $\sigma'$ ). Since small and large items are treated separately by  $\text{HA}_{K+1}$ , the cost of  $\text{HA}_{K+1}$  for serving  $\sigma'_i$  is also  $l_i + \text{NF}(W_i)$ . Hence,  $\text{HM}_K(\sigma_i) = \text{HA}_{K+1}(\sigma'_i)$ . Taking the sum over all classes, we get  $\text{HM}_K(\sigma) = \text{HA}_{K+1}(\sigma')$ . On the other hand, by Lemma 1,  $\text{HA}$  is monotone and  $\text{HA}_{K+1}(\sigma') \leq \text{HA}_{K+1}(\sigma)$ . Consequently,  $\text{HM}_K(\sigma) \leq \text{HA}_{K+1}(\sigma)$ .  $\square$

We show that the upper bound given in the above theorem is tight. Consequently, we have:

**Corollary 1.** *The competitive ratio of  $\text{HM}_K$  is equal to that of  $\text{HA}_{K+1}$ , i.e., it converges to  $T_\infty \approx 1.691$  for large values of  $K$ .*

*Proof.* Let  $\alpha$  denote a lower bound for the competitive ratio of  $\text{HA}_{K+1}$  and consider a sequence  $\sigma$  for which the cost of  $\text{HA}_{K+1}$  is  $\alpha$  times more than that of  $\text{OPT}$ . Define a sequence  $\sigma_\pi$  as a permutation of  $\sigma$  in which items are sorted in increasing order of their sizes. When applying  $\text{HM}_K$  on  $\sigma_\pi$ , all large items will be unmatched in their bins (no other item is packed in their bins). Hence, the cost of  $\text{HM}_K$  for packing  $\sigma_\pi$  is the same as  $\text{HA}_{K+1}$  for packing  $\sigma$ , i.e.,  $\alpha$  times the cost of  $\text{OPT}$  for serving  $\sigma$  and  $\sigma_\pi$  (note that  $\text{OPT}$  uses an identical packing for both  $\sigma$  and  $\sigma_\pi$ ).  $\square$

To achieve a competitive ratio better than 1.7 of  $\text{BF}$  and  $\text{FF}$  for  $\text{HM}_K$ , it is sufficient to have  $K \geq 6$ . In that case,  $\text{HM}_6$  performs as well as  $\text{HA}_7$ , which has a competitive ratio of at most 1.695.

## 2.2 Average-Case Analysis

In this section, we study the average-case performance of the  $\text{HM}$  algorithm under a uniform distribution. Like most related work, we make use of the results related to the *up-right matching* problem. An instance of this problem includes  $n$  points generated uniformly at random in a unit-square in the plane. Each point receives a  $\oplus$  or  $\ominus$  label with an equal probability. The goal is to find a maximum matching of  $\oplus$  points with  $\ominus$  points so that in each pair of matched points the  $\oplus$  point appear above and to the right of the  $\ominus$  point. Let  $U_n$  denote the number of unmatched points in an optimal up-right matching of  $n$  points. For the expected size of  $U_n$ , it is known that  $E[U_n] = \Theta(\sqrt{n} \log^{3/4} n)$  [29, 23, 27, 11]. Given an instance of bin packing defined by a sequence  $\sigma$ , one can make an instance of up-right matching as follows [21]: Each item  $x$  of  $\sigma$  is plotted as a point in the unit square. the vertical coordinate of such point corresponds to the index of  $x$  in  $\sigma$  (normalized to fit in the square). If  $x$  is smaller than  $1/2$ , the point associated with  $x$  is labeled as  $\ominus$  and its horizontal coordinate will be  $2x$ ; otherwise, the point will be  $\oplus$  and its horizontal coordinate will be  $2 - 2x$ . Note that the resulted point will be bounded in the unit square. A solution to the up-right matching instance gives a packing of  $\sigma$  in which the items associated with a pair of matched points are placed in the

same bin. Note that the sum of the sizes of these two items is no more than the bin capacity. Also, in such solution, each bin contains at most two items.

For our purposes, we study  $\sigma_t$  as a subsequence of  $\sigma$  which only includes items which belong to the same class in the HM algorithm. The items in  $\sigma_t$  are generated uniformly at random from  $(\frac{1}{t+1}, \frac{1}{t}] \cup (\frac{t-1}{t}, \frac{t}{t+1}]$ . Since the two intervals have the same length, the items can be plotted in a similar manner on the unit square as follows. The horizontal coordinate of a small item with size  $x$  is  $x \times t(t+1) - t$  and for large items it is  $x \times t(t+1) - (t^2 - 1)$ . The label of the item and its vertical coordinate are defined as before.

Any bin packing algorithm which closes a bin after placing a small item can be applied to the up-right matching problem. Each edge in the up-right matching instance corresponds to a bin which includes one small and one large item. Recall that the algorithm Matching Best Fit (MBF) applies a Best Fit strategy except that it closes a bin as soon as it receives an item with size smaller than or equal to  $1/2$ . So, MBF can be applied in the up-right matching problem. Indeed, it creates an optimal up-right matching, i.e., if we apply MBF on a sequence  $\sigma_t$  which is randomly generated from  $(0, 1]$ , the number of unmatched points will be  $\Theta(\sqrt{n_t} \log^{3/4} n_t)$ , where  $n_t$  is the length of  $\sigma_t$  [29]. We show the same result holds for the bin packing sequences in which items are taken uniformly at random from  $(\frac{1}{t+1}, \frac{1}{t}] \cup (\frac{t-1}{t}, \frac{t}{t+1}]$ :

**Lemma 2.** *For a sequence  $\sigma_t$  of length  $n_t$  in which item sizes are selected uniformly at random from  $(\frac{1}{t+1}, \frac{1}{t}] \cup (\frac{t-1}{t}, \frac{t}{t+1}]$ , we have  $E[\text{MBF}(\sigma_t)] = n_t/2 + \Theta(\sqrt{n_t} \log^{3/4} n_t)$ .*

*Proof.* Define an instance for up-right matching from  $\sigma_t$  as follows: Let  $x$  be the  $i$ th item of  $\sigma_t$  ( $1 \leq i \leq n_t$ ). If  $x$  is small, plot a point with  $\ominus$  label at position  $(x \times t(t+1) - t, t/n)$ ; otherwise, plot a point with  $\oplus$  label at position  $(x \times t(t+1) - (t^2 - 1), t/n)$ . This way, the points will be bounded in the unit square. Since the item sizes are generated uniformly at random from the two intervals and the sizes of the intervals are the same, the point locations and labels are assigned uniformly at random. As a result, the number of unmatched points in the up-right matching solution by MBF is expected to be  $\Theta(\sqrt{n_t} \log^{3/4} n_t)$ . The unmatched points are associated with the items in  $\sigma_t$  which are packed as a single item in their bins by MBF. Let  $s$  denote the number of such items, hence  $E[s] \in \Theta(\sqrt{n_t} \log^{3/4} n_t)$ . Except these  $s$  items, other items are packed with exactly one other item in the same bin. So we have  $\text{MBF}(\sigma_t) - s/2 = n_t/2$  which implies  $E[\text{MBF}(\sigma_t)] = n_t/2 + E[s]/2$ . Since  $E[s] \in \Theta(\sqrt{n_t} \log^{3/4} n_t)$ , the statement of the lemma follows.  $\square$

**Lemma 3.** *For any instance  $\sigma$  of the bin packing problem, the cost of ROM for serving  $\sigma$  is no more than that of MBF.*

*Proof.* Both ROM and MBF open a new bin for each large item. Also, they treat small items which have companions in the same way, i.e., they place the item in the bin of the largest companion and close that bin. The only difference between ROM and MBF is in placing small items without companions where ROM applies the NF strategy while MBF opens a new bin for each item. Trivially, ROM does not open more bins than MBF for these items.  $\square$

**Lemma 4.** *Removing an item does not increase the cost of MBF.*

*Proof.* Let  $\sigma$  denote an input sequence and  $n$  denote the length of  $\sigma$ . We use a reverse induction to show that removing the  $(n-i)$ th item ( $0 \leq i \leq n-1$ ) does not increase the cost of MBF for serving  $\sigma$ . Note that removing the last item does not increase the cost of any algorithm and the base of induction holds. Assume the statement holds for  $i = k+1$ , i.e., removing any item from index  $i \geq k+1$  does not increase the cost of MBF. We show the same holds for  $i = k$ . Let  $n_l$  denote the number of large items in  $\sigma$  and  $n_{ss}$  denote the number of *single small* items, which are the small items which have no companion. For placing a single small item, MBF opens a bin and closes the bin right after placing the item. For the cost of MBF for serving  $\sigma$  we have  $\text{MBF}(\sigma) = n_l + n_{ss}$ . Let  $x$  denote the  $k$ th item in  $\sigma$ . We show removing  $x$  does not increase the cost of MBF. There are a few cases to consider.

First, note that if  $x$  is a small single item, removing it decreases the cost of MBF by one unit. Since the packing of other items does not change, the inductive step trivially holds. Next, assume  $x$  is a small item which has a companion. Removing  $x$  might create a space for another small item  $x'$  in the bin of  $x$ . In case such an item does not exist (i.e., no other item replaces  $x$  in its bin), the packing and consequently the cost of the algorithm do not change. Otherwise,  $x'$  is placed in the bin which includes the companion of  $x$  and closes that bin. Let  $k'$  denote the index of  $x'$  in the sequence and note that  $k' > k$ . Also, let  $\sigma^{-a}$  denote a copy of  $\sigma$  from which an item  $a$  is removed. We have  $\text{MBF}(\sigma^{-x}) = \text{MBF}(\sigma^{-x'})$ , i.e., removing item  $x$  changes the cost of MBF in the same way that removing  $x'$  does. By the induction hypothesis, removing  $x'$  does not increase the cost of MBF and we are done.

The only remaining case is when  $x$  is a large item. If  $x$  does not have a companion, removing  $x$  decreases the cost by one unit and we are done. Now assume  $x$  has a companion  $x'$  and let  $\sigma^{--}$  denote the same sequence as  $\sigma$  in which both  $x$  and  $x'$  are removed. As before, let  $\sigma^{-x}$  denote a copy of  $\sigma$  in which  $x$  is removed. We have  $\text{MBF}(\sigma^{--}) = \text{MBF}(\sigma) - 1$ . On the other hand,  $\text{MBF}(\sigma^{-x}) \leq \text{MBF}(\sigma^{--}) + 1$ . This is because adding a small item to a sequence does not increase the cost of MBF by more than one unit; this holds because MBF closes a bin as soon as a small item is placed in the bin. We conclude that  $\text{MBF}(\sigma^{-x}) \leq \text{MBF}(\sigma)$ , and the inductive step holds.  $\square$

Provided with the above lemmas, we prove the following theorem.

**Theorem 2.** *Let  $\sigma$  be a sequence of length  $n$  in which item sizes are selected uniformly at random from  $(0, 1]$ . The expected wasted space of HM for packing  $\sigma$  is  $\Theta(\sqrt{n} \log^{3/4} n)$ .*

*Proof.* Let  $\sigma^-$  be a copy of  $\sigma$  in which those items which are placed in mature bins are removed. Let  $\sigma_1^-, \dots, \sigma_K^-$  be the subsequences of  $\sigma^-$  formed by items belonging to different classes of HM. We have:

$$\text{HM}(\sigma) = \sum_{t=1}^K \text{ROM}(\sigma_t^-) \leq \sum_{t=1}^K \text{MBF}(\sigma_t^-) \leq \sum_{t=1}^K \text{MBF}(\sigma_t)$$

The inequalities come from Lemmas 3 and 4, respectively. Consequently, by Lemma 2, we have:

$$E[\text{HM}(\sigma)] \leq \sum_{t=1}^K \left( n_t/2 + \Theta(\sqrt{n_t} \log^{3/4} n_t) \right) = \frac{n}{2} + \Theta(\sqrt{n} \log^{3/4} n)$$

Note that the last equation only holds when  $K$  is a constant. The expected cost of OPT is no better than  $n/2$  since half of items are expected to be larger than  $1/2$ . Consequently, we have:

$$E[\text{HM}(\sigma) - \text{OPT}(\sigma)] \in \Theta(\sqrt{n} \log^{3/4} n)$$

$\square$

It should be mentioned that, although the expected waste of HM algorithms is  $\Theta(\sqrt{n} \log^{3/4} n)$ , there is a multiplicative constant involved in the expression which is a function of  $K$ . This implies that the rate of convergence to BF is slower for larger values of  $K$ .

### 3 Refined Harmonic Match

In this section, we introduce a slightly more complicated algorithm, called Refined Harmonic Match (RHM), which has a better competitive ratio than BF, FF, and HM while performing as well as them on average. In introducing RHM, we have been inspired by the Refined Harmonic algorithm of [22].



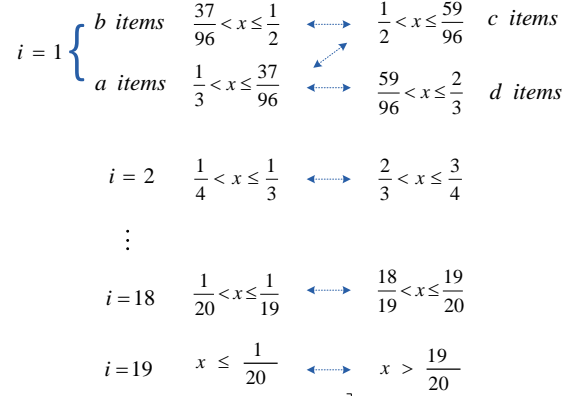


Figure 2: The classes defined by RHM. The algorithm matches items from intervals indicated by arrows.

Similar to HM, RHM divides items into a constant number of classes and treat items of each class separately. The classes defined for RHM are the same as those of  $HM_K$  with  $K = 19$ . The items which belong to class  $k \geq 2$  are treated using HM strategy, i.e., a set of mature bins are maintained. If an item fits in mature bins, it is placed there using BF strategy; otherwise, it is placed together with similar items of its class using ROM strategy. At the same time, the bins closed by ROM subroutine are declared as being mature.

The only difference between HM and RHM in packing items of class 1, i.e., items in range  $(1/3, 2/3]$ . RHM divides items in this range into four groups  $a = (1/3, 37/96]$ ,  $b = (37/96, 1/2]$ ,  $c = (1/2, 59/96]$ , and  $d = (59/96, 2/3]$  (see Figure 2). Similar to Refined Harmonic, to handle the bad sequences which result in lower bound of  $T_\infty$  for competitive ratios of HA and MH, RHM designates a fraction of bins opened by items of group  $a$  to host future  $c$  items. Note that the total size of a  $c$  item and an  $a$  item is no more than 1. However, to ensure a good average-case performance, RHM should be more elaborate than Refined Harmonic. This is because it cannot treat  $b$  apart from other items using a strategy like NF as Refined Harmonic does. In what follows, we introduce an online algorithm called Refined Relaxed Online Match (RRM) as a subroutine of RHM that is specifically used for placing items of class 1.

To place an item  $x$  of the first class ( $x \in (1/3, 2/3]$ ), RRM uses the following strategy. At each step of the algorithm, when two items of the first class are placed in the same bin, that bin is declared as being mature and will be used for placing small items of other classes. More precisely, it will be added to the set of mature bins maintained by the HM algorithm applied for placing items in other classes. If  $x$  is a  $d$ -item, RRM opens a new bin for  $x$ . If  $x$  is a  $c$  item, the algorithm checks whether there are bins which include a single  $a$  item and are designated to have a  $c$  item; in case there are such bins,  $x$  is placed in one of those using BF strategy. In case there is no such a bin, a new bin is opened for  $x$ .

For  $a$  and  $b$  items (small items of class 1), RRM uses the BF strategy to select a bin with enough space which includes a single large item (if there is such a bin). This is particularly important to guarantee a good average-case behavior. If  $x$  is a  $b$  item, the algorithm checks the bin with the highest level in which  $x$  fits; if such a bin includes a  $c$  or a  $b$  item,  $x$  is placed there. Otherwise (when the selected bin does not exist or when it has an  $a$  item), a new bin is opened for  $x$ . If  $x$  is an  $a$  item, the algorithm uses BF strategy to place it into a bin with a  $d$  or  $c$  item. If no suitable bin exist,  $x$  is placed into a bin with a single  $a$  item (there is at most one such bin); if there is no such bin, a new bin is opened for  $x$ .

When a new bin is opened for an  $a$ -item, the bin will be designated to either include a  $c$  item or another  $a$  item in the future. We define *red bins* as those which include two  $a$  items, or a single  $a$  item designated to be paired with another  $a$  item, and define *blue bins* as those which include either a  $c$  item together with an  $a$  or a  $b$  item, or a single  $a$  item designated to be paired with a  $c$  in future. When opening a new bin for an  $a$  item, RHM tries to maintain the number of red bins as close to three times the number of blue bins as

possible. Namely, if the number of red bins is less than 3 times of blue bins, it declares the opened bin as a red bin to host another  $a$  item in future; otherwise, the new open bin is declared as a blue bin to host a  $c$  item in future. This way, the number of red bins is close to (but never more than) three times of blue bins. Note that, when many  $b$  items are placed together with  $c$  items, the resulting bins will be blue. In this case, the algorithm does not limit the number of blue bins unless it opens bins for  $a$  items. Consequently, the number of red bins can be less than three times of blue bins. Algorithm 1 illustrates how RRM works.

---

**Algorithm 1:** RRM algorithm: Placing a sequence of items in range  $(1/3, 2/3]$

---

**input:** A sequence  $\sigma = \langle \sigma_1, \sigma_2, \dots, \sigma_n \rangle$  of items in range  $(1/3, 2/3]$

$N_{a_1}, N_{a_2}, N_{aa}, N_{ab}, N_{ac}, N_b, N_{bc} \leftarrow 0$

**for**  $i \leftarrow 1$  **to**  $n$  **do**

**switch**  $\sigma_i$  **do**

**case**  $d$  **item:**

            | open a new bin for  $\sigma_i$

**case**  $c$  **item:**

            | **if**  $N_{a_1} > 0$  **then**

                | Use BF to place  $\sigma_i$  in a bin with an  $a$  item  $N_{a_1} \leftarrow N_{a_1} - 1$ ;  $N_{ac} \leftarrow N_{ac} + 1$

                |  $\{N_{a_1}$  is the number of bins with a single  $a$  item which are designated to have a  $c$  item $\}$

            | **else** open a new bin for  $\sigma_i$ ;  $N_c \leftarrow N_c + 1$ ;

**case**  $b$  **item:**

            | select the bins with a  $c$  item which have enough capacity for  $\sigma_i$

            | **if there is a selected bin then**

                | place  $\sigma_i$  into the bin with the highest level among the selected bins;

                |  $N_{bc} \leftarrow N_{bc} + 1$ ;  $N_c \leftarrow N_c - 1$

            | **else if**  $N_b = 1$  **then**

                | place  $\sigma_i$  into the bin with a single  $b$  item;  $N_b \leftarrow 0$

            | **else** open a new bin for  $\sigma_i$ ;  $N_b \leftarrow 1$ ;

**case**  $a$  **item:**

            | select the bins with a  $d$  item which have enough capacity for  $\sigma_i$

            | **if there is such a bin then**

                | place  $\sigma_i$  into the bin with the highest level among those bins

            | **else if**  $N_c > 0$  **then**

                | place  $\sigma_i$  into the bin with the largest  $c$  item;  $N_{ac} \leftarrow N_{ac} + 1$ ;  $N_c \leftarrow N_c - 1$

            | **else if**  $N_{a_2} = 1$  **then**

                | place  $\sigma_i$  into any with a single  $a$  item;  $N_{a_2} \leftarrow 0$ ;  $N_{aa} \leftarrow N_{aa} + 1$

            | **else**

                | place  $\sigma_i$  in a new (empty) bin

                |  $\{compare the number of red bins with 3 times number of blue bins\}$

                | **if**  $N_{aa} < 3(N_{ac} + N_{a_1} + N_{bc})$  **then**

                    |  $N_{a_2} \leftarrow 1$ ;  $\{declare the opened bin as a red bin (an  $a_2$ -bin) \}$

                | **else**

                    |  $N_{a_1} \leftarrow N_{a_1} + 1$ ;  $\{declare the opened bin as a blue bin (an  $a_1$ -bin) \}$

**endsw**

**end**

---

### 3.1 Worst-Case Analysis

In this section, we provide an upper bound of 1.636 for the competitive ratio of RRM. We start by introducing some notations. Bins in a packing by RRM can be divided into the following groups:  $d$ -bins which include

a  $d$ -item (might also include an  $a$  item),  $c$ -bins (resp.  $b$ -bins) which include a single  $c$  (resp.  $b$ ) item,  $a_1$ -bins (resp.  $a_2$ -bins) which include a single  $a$  item and are designated to include a  $c$  (resp. an  $a$ ) item in future,  $bb$ -bins (resp.  $aa$ -bins) which include two  $b$  (resp.  $a$ ) items,  $ac$ -bins which include an  $a$  item and a  $c$  item, and  $bc$ -bins which include a  $b$  item and a  $c$  item. Note that there is at most one  $b$ -bin and one  $a_2$ -bin (otherwise, two of those bins form a  $bb$ -bin or  $aa$ -bin, respectively). We use respectively capital  $N$  and lower case  $n$  to refer to the number of bins and items:  $N_\alpha$  denotes the number of bins of type  $\alpha$ , e.g.,  $N_{ac}$  indicates the number of  $ac$ -bins. Similarly,  $N_{red}, N_{blue}$  denote the number of red and blue bins in the packing. Note that  $N_{red} = N_{aa} + N_{a_2}$  and  $N_{blue} = N_{ac} + N_{bc} + N_{a_1}$ . We use  $n_x$  to denote the number of items of type  $x$  ( $x \in \{a, b, c, d\}$ ). Moreover, we use  $n_{b_1}$  to denote the number of  $b$  items which are packed with a  $c$  item ( $n_{b_1} = N_{bc}$ ) and  $n_{b_2}$  to denote the number of other  $b$  items ( $n_{b_1} + n_{b_2} = n_b$ ). Counting the number of  $a$  and  $b_1$  items we get  $n_a + n_{b_1} = 2N_{aa} + N_{ac} + N_{a_1} + N_{a_2} + N_{bc}$ . Since  $N_{a_2} \leq 1$ , by definition of red and blue bins, we get

$$n_a + n_{b_1} - 1 \leq 2N_{red} + N_{blue} \leq n_a + n_{b_1} \quad (1)$$

We refer to the above equation in a few places in our analysis. Since RHM uses the same strategy as HM for placing items in classes  $k \geq 2$ , and HM never opens more bins than HA does (Theorem 1), we can prove the following lemma.

**Lemma 5.** *The cost of RHM for serving items of classes  $k \geq 2$  is upper bounded by*

$$\text{RHM}(\sigma) \leq \text{RRM}(\sigma_{cl_1}) + \sum_{k=2}^{18} \frac{n_k}{k+1} + 20W'/19 + 20$$

in which  $\sigma_{cl_1}$  is the subsequence formed by items of class 1,  $n_k$  is the number of items in class  $k$ , and  $W'$  is the total size of items in class 19 (the last class).

*Proof.* Since RHM performs similarly to HM for placing items of class  $k \geq 2$ , the proof of Theorem 1 can be applied to state that RHM does not open more bins than Harmonic algorithm does for placing these items. Harmonic places  $k+1$  items of class  $k$  in the same bin ( $2 \leq k \leq 18$ ); hence, it opens at most  $\frac{n_k}{k+1} + 1$  bins for items in such class. The empty space in any bin assigned to items of the last class (class 19) is at most  $1/20$  since the size of items in this class is no more than  $1/20$ . Hence, the number of opened bins for this class is at most  $20W'/19 + 1$ . Note that some items in classes  $k \geq 2$  might be placed in the mature bins maintained by HM (including the bins released by RRM). When comparing with Harmonic algorithm, we can think of these items as being removed from the Harmonic packing. Since the Harmonic is monotone (Lemma 1), removing these items does not increase the number of bins. Hence, the claimed upper bound still holds.  $\square$

**Theorem 3.** *The competitive ratio of RHM is at most  $373/228 < 1.636$ .*

*Proof.* RRM is defined in a way that no  $c$ -bin and  $a_1$ -bin can be open at the same time. We consider the following two cases based on the final packing of RRM for the subsequence formed by items of type 1:

- Case 1: there is no  $c$ -bin in the final packing, while there is at least one  $a_1$ -bin in the packing.
- Case 2: there is no  $a_1$ -bin in the final packing.

We prove the theorem for these two cases separately.

**Case 1:** Assume there is no  $c$ -bin in the final packing, while there is at least one  $a_1$ -bin in the packing. Let  $x$  be the last  $a$  item for which an  $a_1$ -bin is opened. We claim that no blue bin is added to the packing after placing  $x$ . Blue bins are opened by  $a$  or  $c$  items. A new blue bin cannot be opened by a  $c$  item as such a  $c$  item should have been placed in one of the existing  $a_1$  bins. Also, a new blue bin cannot be opened by an  $a$  item since that results in a bin with a single  $a$  item; this contradicts  $x$  being the last item for which an  $a_1$  bin is opened. So, the number of blue bins does not increase after placing  $x$ . At the time of placing  $x$ , the number of red bins is no less than three times the number of blue bins; otherwise, the bin opened for  $x$  would have been declared as a red bin (i.e., an  $a_2$ -bin). So, we have  $-3 \leq N_{red} - 3N_{blue} \leq 3$ . Using Equation 1 we get:

$$N_{red} + N_{blue} \leq 4n_a/7 + 4n_{b_1}/7 + 11/7$$

And for the total cost of RRM we will have:

$$\begin{aligned} \text{RRM}(\sigma_{cl_1}) &= N_d + N_{ac} + N_{bc} + N_b + N_{bb} + N_a + N_{a'} + N_{aa} \\ &\leq n_d + n_{b_2}/2 + N_{blue} + N_{red} \\ &\leq n_d + n_{b_2}/2 + 4n_a/7 + 4n_{b_1}/7 + 11/7 \\ &\leq n_d + 4n_b/7 + 4n_a/7 + 11/7 \end{aligned}$$

Plugging this into the upper bound given by Lemma 5, we get:

$$\text{RHM}(\sigma) \leq \sum_{k=2}^{19} \frac{n_k}{k+1} + 20W'/19 + n_d + 4n_b/7 + 4n_a/7 + 22 \quad (2)$$

To further analyze the algorithm, we use a weighting function similar to that of [22]. We define a weight for each item so that the total weight of items in a sequence, denoted by  $W(\sigma)$ , becomes an upper bound for the cost of RHM. At the same time, we show that the total weight of any set of items which fit in a bin is at most 1.63; this implies that the cost of OPT for serving  $\sigma$  is at least  $W(\sigma)/1.63$ . Consequently, the ratio between the costs of RHM and OPT is at most 1.63.

We define a weight for each item in the following manner. Small items in class  $k$  ( $2 \leq k \leq 18$ ) have weight  $1/(k+1)$ . The weight of a small item  $x$  of class 19 is  $20x/19$ . The weight of  $d$  items and items larger than  $2/3$  is 1; the weight of  $c$  item is 0, and the weight of  $b$  and  $a$  items is  $4/7$ . This way, as the above Inequality 3.1 suggests, the total weight of items in a sequence is an upper bound for the cost of RHM. Next, we study the maximum weight of items in a bin of OPT. Let  $\beta_1, \beta_2, \dots, \beta_t$  denote the set of items in a bin of OPT so that  $\beta_1 \geq \beta_2 \geq \dots \geq \beta_t$ . Let  $W_{opt}$  denote the total weight of items in such a bin, i.e.,  $W_{opt} = \sum_{i=1}^t \omega(\beta_i)$ . We claim that  $W_{opt} < 1.63$ , which implies the competitive ratio is upper bounded by  $W_{opt}$ . The *density* of any item, defined as the ratio between the weight and the size of the item, is at most  $\frac{4/7}{1/3} = 12/7 < 1.72$  for  $b$ -items. For  $a$  items, the density is at most  $\frac{4/7}{37/96} < 1.48$ . For items smaller than  $b$  items, the density decreases as the class increases (from at most  $4/3$  for items of class 2 to at most  $20/19$  for items in classes 18 and 19). The density of large items is at most  $96/59 < 1.63$ .

To prove the claim, we do a case analysis on the value of  $\beta_1$ : I) assume  $\beta_1$  has a size larger than  $59/96$ , i.e., it is a  $d$  item or larger. If  $\beta_2$  is an  $a$  or a  $b$  item, we will have  $\beta_3 + \dots + \beta_t \leq 5/96 < 1/19$ . So, all other items belong to classes 18 or 19 and their density is at most  $20/19$ . Hence, the total weight of items in the bin will be at most  $1 + 4/7 + 5/96 \times 20/19 < 1.63$ . If  $\beta_2$  is smaller than or equal to  $1/3$ , the total size of all items except  $\beta_1$  is at most  $37/96$  and their density is at most  $4/3$ ; the total weight will be  $1 + 37/96 \times 4/3 < 1.52$ . II) Assume  $\beta_1$  is a  $c$  item, i.e., its weight is 0. The total sizes of other items in the bin (all items except  $\beta_1$ ) is at most  $1/2$  and their density is upper bounded by  $12/7$ . Hence, the total weight of items in the bin will be  $1/2 \times 12/7 < 1$ . III) Assume  $\beta_1$  is a  $b$  item. Assume  $\beta_2$  is a  $b$  or an  $a$  item; so, the total size of all other items is at most  $27/96$ , while their density is at most  $4/3$  (note that they belong to class 2 or higher). Hence, the total weight of items in the bin will be  $4/7 + 4/7 + 27/96 \times 4/3 < 1.52$ . Next, assume  $\beta_2$  is smaller than  $a$  items; the total size of all items (except  $\beta_1$ ) will be at most  $59/96$  while their weight is at most  $4/3$ . The total weight of items in a bin will be at most  $4/7 + 59/96 \times 4/3 < 1.4$ . IV) Assume  $\beta_1$  is an  $a$  item. Assume  $\beta_2$  is also an  $a$  item; so, the total size of all other items is at most  $1/3$ , while having a density of at most  $4/3$  (note that they belong to class 2 or higher). Hence, the total weight of items in the bin will be  $4/7 + 4/7 + 1/3 \times 4/3 < 1.59$ . Next, assume  $\beta_2$  is smaller than  $a$  items; the total size of all items (except  $\beta_1$ ) will be at most  $2/3$  while their weight is at most  $4/3$ . The total weight of items in a bin will be at most  $4/7 + 2/3 \times 4/3 < 1.46$ . V) Finally, consider  $\beta_1$  is smaller than  $1/3$ ; hence, the density of all items is at most  $4/3$ . Consequently, the total weight of items in the bin is at most  $4/3$ .

To summarize, the total cost of RHM for serving a sequence  $\sigma$  is no more than the total weight of items in  $\sigma$ , denoted by  $W(\sigma)$ . At the same time, the total weight of items in a bin by OPT is at most 1.63 which implies that the cost of OPT for serving  $\sigma$  is at least  $W(\sigma)/1.59$ . We conclude that the competitive ratio of RHM is at most 1.59 in this case.

**Case 2:** Assume there is no  $a_1$ -bin in the packing. For the cost of RRM for serving  $\sigma_{cl_1}$ , we have:

$$\begin{aligned} \sigma_{cl_1} &= N_d + N_c + N_{ac} + N_{bc} + N_{bb} + N_{aa} + N_{a_2} \\ &\leq n_d + n_c + n_{b_2}/2 + N_{red} + 1 \end{aligned}$$

Recall that the algorithm ensures that  $N_{red} \leq 3N_{blue} + 3$ . By Equation 1, we get  $N_{red} \leq 3n_a/7 + 3n_{b_1}/7 + 3/7$ . Plugging this into the above inequality, we will get:

$$\begin{aligned} \text{RRM}(\sigma_{cl_1}) &\leq n_d + n_c + n_{b_2}/2 + 3n_a/7 + 3n_{b_1}/7 + 2 \\ &< n_d + n_c + n_b/2 + 3n_a/7 + 2. \end{aligned}$$

By Lemma 5, for the total cost of RHM, we will have:

$$\text{RHM}(\sigma) \leq \sum_{k=2}^{18} \frac{n_k}{k+1} + 20W'/19 + n_d + n_c + n_b/2 + 3n_a/7 + 23 \quad (3)$$

Similar to Case 1, we use a weighting technique. For all items, except for  $a$ ,  $b$ , and  $c$  items, the weights are defined similar to Case 1. For  $a$ ,  $b$ , and  $c$  items, the weights are respectively  $3/7$ ,  $1/2$ , and 1. As Inequality 3.1 suggests, this definition for weights ensures that the total weight of items is an upper bound for the cost of RHM. As before, we study the maximum weight of a bin in the packing of OPT; let  $\beta_1 \geq \beta_2 \geq \dots \geq \beta_t$  be the items in such a bin and  $W_{opt}$  be their total weight. We claim that  $W_{opt} < 1.63$ . Note

that the density of  $d$  items and larger items are at most  $96/59 < 1.63$ , the density of  $c$ ,  $b$  and  $a$  items are respectively upper bounded by 2, 1.3, and 1.29, and the density of items smaller than  $1/3$  which belongs to class  $k \geq 2$  is at most  $\frac{k+2}{k+1}$  except the last class (class 19) for which the density is at most  $20/19$ .

We do a case analysis as before. First, assume  $\beta_1$  is not a  $c$ -item. In this case, the density of all items, and consecutively their total weight, is less than 1.63. Next, assume  $\beta_1$  is a  $c$  item. Now, if  $\beta_2$  is a  $b$  item, the total weight of other items will be at most  $11/96 < 1/8$ . Hence, these items belong to class 7 or higher and their density is at most  $9/8$ . The total weight of items in the bin will be  $1 + 1/2 + 11/96 \times 9/8 \leq 1.62$ . Next, assume  $\beta_2$  is an  $a$  item; the total weight of other items will be at most  $1/6$ . These items belong to class 5 or higher and their density is at most  $7/6$ . Hence, the total weight of items will be at most  $1 + 3/7 + 1/6 \times 7/6 < 1.63$ . Next, assume  $\beta_2$  belongs to class 2; the total weight of other items will be at most  $1/4$ . Now, if  $\beta_3$  belongs to class 3, the weight of other items will be at most  $1 - 1/2 - 1/4 - 1/5 = 1/20$ ; so they belong to the last class and their density is at most  $20/19$ . The total weight of items will be at most  $1 + 1/3 + 1/4 + 1/20 \times 20/19 = 373/228 \approx 1.636$ . If  $\beta_3$  belongs to class 4 or higher, its density will be at most  $6/5$ , and the total weight of items in the bin will be at most  $1 + 1/3 + 1/4 \times 6/5 < 1.634$ . Finally, assume  $\beta_2$  belongs to class 3 or higher; so, all items except  $\beta_1$  have a density of at most  $5/4$ . The total weight of items will be at most  $1 + 1/2 \times 5/4 = 1.625$ .

To summarize, the total weight of items in a bin in OPT's packing is at most  $373/228$  which implies that the cost of OPT for serving  $\sigma$  is at least  $228W(\sigma)/373$ . Recall that the cost of RHM for serving  $\sigma$  is upper bounded by  $W(\sigma)$ . We conclude that the competitive ratio of RHM is at most  $373/228$  in this case.  $\square$

### 3.2 Average-Case Analysis

We show that the average-case performance of RHM is as good as BF, FF, and HM. Except the following lemma, other aspects of the proof are similar to those in Section 2.2.

**Lemma 6.** *For any instance  $\sigma$  of the bin packing problem in which items are in range  $(1/3, 2/3]$ , the cost of RRM for serving  $\sigma$  is no more than that of Matching Best Fit (MBF).*

*Proof.* The key observation is that RRM uses BF strategy to place a small item  $x$  in a bin which includes a large item; and only if such a bin does not exist, it deviates from the BF strategy. Let  $S_{RRM}^t$  and  $S_{MBF}^t$  respectively denote the set of large items which are not accompanied by a small item in the packings maintained by RRM and MBF (respectively) after placing the first  $t$  items in  $\sigma$  ( $0 \leq t \leq n$ , where  $n$  is the length of  $\sigma$ ); we refer to these sets as *single-sets* of the algorithms. We claim that for all values of  $t$ , a single-set of RRM is a subset of that of MBF, i.e.,  $S_{RRM}^t \subset S_{MBF}^t$ . We prove this by induction. Note that for  $t = 0$  both single-sets are empty and the base case holds. Assume  $S_{RRM}^t \subset S_{MBF}^t$  for some  $t > 0$  and let  $x$  denote the  $(t + 1)$ th item in  $\sigma$ . If  $x$  is a large item, MBF opens a bin for  $x$ , and  $x$  will be included in the single-set for MBF;  $x$  may or may not be added to the single-set of RRM (it will not be added if it is a  $c$  item and there are  $a_1$  bins in the packing). Regardless, we will have  $S_{RRM}^{t+1} \subset S_{MBF}^{t+1}$ . Next, assume  $x$  is a small item. MBF and RRM both use BF strategy to place  $x$  in one of the bins in  $S_{MBF}^t$  and  $S_{RRM}^t$ . If such a bin does not exist for MBF, by induction hypothesis, it will not exist for RRM and the induction statement holds. Next, assume MBF places  $x$  in a bin  $B \in S_{MBF}^t$ ; so,  $B$  will be removed from single-set of MBF, i.e., we have  $S_{MBF}^{t+1} = S_{MBF}^t - \{B\}$ . If  $B \notin S_{RRM}^t$ , the induction statement holds because  $S_{RRM}^{t+1}$  will be a subset of  $S_{RRM}^t$  which is a indeed a subset of  $S_{MBF}^{t+1}$  (a non-common items is removed from  $S_{MBF}^t$ ). If  $B \in S_{RRM}^t$ , RRM places  $x$  in  $B$ ; this is because, similar to MBF, RRM uses BF strategy to place small items in bins which include large items. Consecutively, we have  $S_{RRM}^n \subset S_{MBF}^n$ .

The cost of MBF for serving  $\sigma$  is  $n_{small} + |S_{MBF}^n|$  in which  $n_{small}$  is the number of small items in  $\sigma$ . This is because MBF opens a new bin for each small item. In the final packing of RRM, the number of bins which include small items is no more than  $n_{small}$ . Other bins in the packing are associated with items in

$S_{RRM}^n$ ; since the single-set of RRM is a subset of that of single-set of MBF, we have  $|S_{RRM}^n| \leq |S_{MBF}^n|$ . Hence, in total, the number of bins in the packing of RRM is no more than that of MBF.  $\square$

**Theorem 4.** *Let  $\sigma$  be a sequence of length  $n$  in which item sizes are selected uniformly at random from  $(0, 1]$ . The expected wasted space of RHM for packing  $\sigma$  is  $\Theta(\sqrt{n} \log^{3/4} n)$ .*

*Proof.* Let  $\sigma^-$  be a copy of  $\sigma$  in which those items which are placed in mature bins are removed. Also, let  $\sigma_2^-, \dots, \sigma_{19}^-$  be the subsequences of  $\sigma^-$  formed by items belonging to different classes of HM. We have

$$\text{HM}(\sigma) = \text{RRM}(\sigma_1) + \sum_{t=2}^{19} \text{ROM}(\sigma_t^-) \leq \sum_{t=1}^{19} \text{MBF}(\sigma_t^-) \leq \sum_{t=1}^{19} \text{MBF}(\sigma_t)$$

The second-to-last inequality comes from Lemmas 3 and 6 and the last inequality comes from Lemma 4. Consequently, by Lemma 2, we have:

$$E[\text{HM}(\sigma)] \leq \sum_{t=1}^{19} \left( n_t/2 + \Theta(\sqrt{n_t} \log^{3/4} n_t) \right) = \frac{n}{2} + \Theta(\sqrt{n} \log^{3/4} n)$$

The expected cost of OPT is  $n/2$  (it is not better since half items are expected to be larger than  $1/2$ ). Consequently,  $E[\text{HM}(\sigma) - \text{OPT}(\sigma)] = \Theta(\sqrt{n} \log^{3/4} n)$  which completes the proof.  $\square$

## 4 Experimental Evaluation

The results of the previous sections indicate that HM and RHM have similar average-case performance as BF if we assume a uniform, *continuous* distribution for item sizes. In this section, we further observe the performance of these algorithms on sequences which follow other distributions. In doing so, we experimentally compare HM and RHM against classical bin packing algorithms. Table 2 gives details of the datasets that we generated for our experiments. In all cases,  $E$  indicates the size of the bins. For each set-instance, the item sizes are randomly taken from a subset of the set  $\{1, 2, \dots, E\}$  of integers; this subset defines the *range* of the items in the set-instance. Typically, we have  $E = 1000$ , and the range is  $[1, 1000]$ . Here, we briefly describe the considered distributions:

- **Discrete Uniform Distribution (DU sequences):** We test the algorithms on *discrete* uniform distributions. It is known that average-case behavior of bin packing algorithms can be substantially different under discrete and continuous distributions [6]. The bin packing problem is extensively studied under discrete uniform distributions (see, e.g., [6, 17, 1]).
- **NORMAL and POISSON sequences:** Normal and Poisson distributions are two natural alternatives for generating item sizes. Both of these distributions are previously studied for generating bin packing sequences (see, e.g., [19, 31]).
- **Zipfian Distribution (ZD sequences):** In bin packing sequences that follow the Zipfian distribution, item sizes follow the power-law, i.e., a large number of items are pretty small while a small number of items are quite large. The distribution has a parameter  $\theta$  ( $0 < \theta < 1$ ) that indicates how skewed the distribution is. Bin packing sequences with Zipfian distribution are considered in the experiments in [2].
- **SORTD sequences:** To create an instance of this family, we take a sequence of uniformly randomized items and sort it in decreasing order of item sizes. This way, we can compare the performance of *offline* versions of the algorithms (where sequences are sorted in decreasing order before being packed in an online manner).

Set-instance	Distribution	$E$	Range
DU0	Uniform	100	(1,E)
DU1	Uniform	500	(1,E)
DU2	Uniform	1,000	(1,E)
DU3	Uniform	1,000	(1,E/2)
DU4	Uniform	1,000	(1,E/10)
NORMAL	Normal ( $\mu = E/2, \sigma = E/6$ )	1,000	(1,E)
POISSON	Poisson ( $\lambda = E/3$ )	1,000	(1,E)
ZIPF1	Zipfian ( $\theta = 1/2$ )	1,000	(1,E)
ZIPF2	Zipfian ( $\theta = 1/3$ )	1,000	(1,E)
SORTD	Uniform, sorted decreasing	1,000	(1,E)
WD1	Weibull ( $k = 0.454$ )	1,000	(1,E)
WD2	Weibull ( $k = 1.044$ )	1,000	(1,E)
BPSD1	BPS distribution ( $s = 100$ )	1,000	(E/4,E/2)
BPSD2	BPS distribution ( $s = 100$ )	1,000	(1,E/4)

Table 2: . The distributions used to create set-instances to compare algorithms.

- Weibull Distribution (WD sequences): In [5], it is shown that Weibull distribution can be used to model real-world bin packing benchmarks. The values considered for the shape parameter  $k$  are among the ones suggested in [5]. The scale parameter of the distribution is set to be proportional to  $E$ .
- Bounded Probability Sampled Distributions (BPS sequences): To get these sequences, a random distribution is generated as follows. Given a parameter  $s$ , we select  $s$  random numbers in a given range and assign random weights to them. The probability associated with an item in the distribution is proportional to its weight. These sequences were first introduced in [18] and later used in the experiments in [2].

For each of the indicated set-instances, we create 1000 random sequences of length  $10^6$  and compute the average costs of different algorithms for packing these sequences. Beside Any-Fit and Harmonic family of algorithms, we also consider *Sum-of-Squares (SS) Algorithm* which performs well for discrete distributions [16, 14, 15]. To place an item into a partial packing  $P$ , SS defines *sum of squares* of  $P$ , denoted by  $ss(P)$ , as  $\sum_h N_P(h)^2$ ; here  $N_P(h)$  denotes the number of bins with level  $h$  in  $P$ . To place an item  $x$ , SS places  $x$  into an existing bin, or opens a new bin for  $x$ , so as to yield the minimum possible value of  $ss(P')$  for the resulting packing  $P'$ . Note that SS is not well-defined for the classical, continuous version of the bin packing problem. In [15], it is proved that for any discrete distribution in which the optimal expected waste is sub-linear, SS also has sub-linear expected waste. In particular, for those distributions where the optimal expected waste is constant (the so-called a *perfect* distributions), SS has an expected waste of at most  $O(\log n)$ . In our experiments, we treat SS as the closest online algorithm to OPT. We note that the competitive ratio of SS is at least 2 and at most 2.77 [14] which is worse than most online algorithms.

Figure 3 shows the average costs (the number of opened bins) of the classical bin packing algorithms, as well as HM and RHM, for serving the above set-instances. For algorithms that classify items by their sizes (e.g., HA and HM), the number of classes (the value of  $K$ ) is set to 20. The results in Figure 3 indicate that in all cases HM and RHM perform significantly better than other members of Harmonic family. At the same time, they have comparable performance with BF and FF. In most cases, HM and RHM perform better than FF, and in some cases, e.g., NORMAL set, they even perform better than BF.



	NF	WF	HA	RFF	RHA	OM	FF	BF	HM	RHM	SS
DU0	665,045	584,597	642,786	643,455	646,392	501,025	502,075	500,918	507,042	507,042	501,116
DU1	666,298	585,521	644,288	644,504	648,057	501,181	502,973	501,069	502,543	502,543	502,357
DU2	666,452	585,638	644,660	644,418	648,376	501,247	503,162	501,133	502,219	502,219	503,332
DU3	298,417	275,248	289,701	289,225	297,146	298,417	251,261	251,148	251,024	260,394	250,015
DU4	51,695	50,912	51,598	50,010	51,598	51,695	50,010	50,009	50,014	50,014	50,005
NORMAL	697,005	604,915	712,149	718,129	718,445	500,245	501,400	500,128	499,922	499,922	501,737
POISSON	405,402	371,058	414,234	454,669	448,730	405,402	343,900	343,869	351,617	392,842	333,800
ZIPF1	514,415	448,155	502,305	500,603	505,940	403,952	392,601	392,219	392,351	392,350	390,266
ZIPF2	460,120	400,057	450,988	448,809	454,470	365,523	352,936	352,799	352,970	352,969	351,275
SORTD	644,200	499,581	644,207	643,800	647,929	499,718	499,565	499,565	500,534	500,534	502,219
WD1	324,191	281,857	320,150	317,761	322,820	263,414	251,137	251,111	251,449	251,449	251,064
WD2	569,643	496,280	557,306	556,296	561,477	443,420	434,802	433,690	432,165	432,164	429,076
BPSD1	435,216	381,285	466,223	465,861	471,811	349,072	335,850	335,552	333,700	333,699	332,999

Figure 3: Average performance of online bin packing algorithms for different set-instances. The indicated numbers for each algorithm represent the average costs of the algorithm for different set-instances. In most cases, there is a gap between the cost of HM (and RHM) and other Harmonic-based algorithms. The data-bar of an algorithm for a set-instance indicates the *rank* of the cost of the algorithm, among the costs of all algorithms, for serving that set-instance.

Comparing the costs of HM and RHM for DU0, DU1, and DU2, we observe that their relative performance improves when the size of the bins (i.e.,  $E$ ) increases. For small value of  $E = 100$  (UD0), these algorithms are slightly worse than FF. However, as  $E$  increases to 1000 (UD2), the algorithms perform better than FF and converge to BF. This is in accordance with the results in Sections 2.2 and 3.2 which imply that, for continuous uniform distribution, the expected costs of HM and RHM converge to that of BF. Note that as  $E$  goes to infinity, the discrete distribution estimates a continuous one.

For symmetric distributions, where items of sizes  $x$  and  $E - x$  appear with the same probability, the expected costs of HM and RHM are equal. A difference between the packings of HM and RHM happens when a number of small items of the first class (items of type  $a$  in RHM) appear before any large item of the same class (an item of type  $c$ ). In these cases, RHM ‘reserves’ some bins for subsequent large items (by declaring the bins as being blue). For symmetric distributions, however, it is unlikely that many small items appear before the next large item. Consequently, as the numbers in Figure 3 reflect, the average costs of HM and RHM are the same for symmetric sequences. On the other hand, for asymmetric sequences where small items are more likely to appear, e.g., DU3 and POISSON, HM has an advantage over RHM. In these sequences, there is no reason to reserve bins for the large items since they are unlikely to appear.

Finally, we note that for SORTD, the costs of HM and RHM are comparable to those of BF and FF. This implies that, on average, the offline versions of these algorithms are comparable with the well-known First-Fit-Decreasing and Best-Fit-Decreasing algorithms. It remains open whether the same statement holds for the worst-case performance of the offline algorithms.

## 5 Discussion

HM and RHM can be seen as variants of Harmonic and Refined Harmonic algorithms in which small and large items are carefully matched in order to improve the average performance, while preserving the worst-case performance. We believe that the same approach can be applied to improve the average performance of other Super Harmonic algorithms, and in particular that of Harmonic++ algorithm (which is currently the best online bin packing algorithm, regarding the competitive ratio). Given the complicated nature of these

algorithm, modifying them involves a detailed analysis which we leave as a future work.

The relative worst order analysis is an alternative method for comparing online algorithms. This method considers the worst ordering of a given sequence for two online algorithms and indicates their costs on these orderings. Then, among all sequences, it considers the one that maximizes the worst-case ratio between the two algorithms (see [4] for a precise definition). It is known that under relative worst order analysis, First Fit is no worse than any Any Fit algorithm (and in particular Best Fit) [4]. Also, Harmonic algorithm is not comparable to FF for sequences which include very small items. However, when all items are larger than  $\frac{1}{K+1}$  ( $K$  is the parameter of the Harmonic algorithm), Harmonic is better than FF by a factor of  $6/5$  [4]. Applying Theorem 1, we conclude that when all items are larger than  $\frac{1}{K+2}$ , Harmonic Match with parameter  $K$  is strictly better than FF and BF under the relative worst order analysis. This provides another evidence for the advantage of Harmonic Match over BF and FF.

Many online bin packing algorithms, e.g., BF and FF, have the undesired property that removing an item might increase the cost of the algorithm [24]. This ‘anomalous’ behavior results in an unstable algorithm which is harder to analyze. As mentioned earlier, an algorithm is called ‘monotone’ if removing an item does not increase its cost. It is not clear whether HM and RHM are monotone; however, a slight twist in HM results in a monotone algorithm. Consider a modified algorithm HMM that works similar to HM except that it does not maintain mature bins, i.e., it closes a bin as soon as it becomes mature. It is not hard to see that HMM is a monotone algorithm. At the same time, the results related to the worst-case and average-case performance of HM hold also for HMM (Corollary 1 and Theorem 2). However, HMM performs slightly worse than HM on discrete distributions evaluated in Section 4. We leave further analysis of monotonous behaviour of this algorithm as a future work.

## References

- [1] Albers, S., Mitzenmacher, M.: Average-case analyses of First Fit and Random Fit bin packing. In: Proc. 9th Symp. on Discrete Algorithms (SODA). pp. 290–299. Society for Industrial and Applied Mathematics (1998)
- [2] Applegate, D., S, L., Buriol, B.L.D., Johnson, D.S., Shor, P.W.: The cutting-stock approach to bin packing: Theory and experiments. In: Proc. 5th Meeting on Algorithm Engineering and Experiments (ALENEX) (2003)
- [3] Bentley, J.L., Johnson, D.S., Leighton, F.T., McGeoch, C.C., McGeoch, L.A.: Some unexpected expected behavior results for bin packing. In: Proc. 16th Symp. on Theory of Computing (STOC). pp. 279–288 (1984)
- [4] Boyar, J., Favrholt, L.M.: The relative worst order ratio for online algorithms. ACM Trans. Algorithms 3(2) (2007)
- [5] Castiñeiras, I., De Cauwer, M., O’Sullivan, B.: Weibull-based benchmarks for bin packing. In: Proc. 18th International Conference on Principles and Practice of Constraint Programming (CP). pp. 207–222. Springer-Verlag (2012)
- [6] Coffman, E.G., Courcoubetis, C.A., Garey, M.R., Johnson, D.S., McGeogh, L.A., Shor, P.W., Weber, R.R., Yannakakis, M.: Fundamental discrepancies between average-case analyses under discrete and continuous distributions - a bin packing case study. In: Proc. 23rd Symp. on Theory of Computing (STOC). pp. 230–240 (1991)
- [7] Coffman, E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: A survey. In: Hochbaum, D. (ed.) Approximation algorithms for NP-hard Problems. PWS Publishing Co. (1997)
- [8] Coffman, E.G., Hofri, M., So, K., Yao, A.C.C.: A stochastic model of bin packing. Inform. and Control 44, 105–115 (1980)
- [9] Coffman, E.G., Johnson, D.S., Shor, P.W., Weber, R.R.: Bin packing with discrete item sizes, part ii: Average case behaviour of First Fit (1996), unpublished manuscript
- [10] Coffman, E.G., Lueker, G.S.: Probabilistic analysis of Packing and Partitioning Algorithms. John Wiley, New York (1991)
- [11] Coffman, E.G., Shor, P.W.: A simple proof of the  $O(\sqrt{n \log^3 4})$  up-right matching bound. SIAM J. Discrete Math. 4, 48–57 (1991)
- [12] Coffman Jr., E.G., Csirik, J., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: survey and classification. In: Pardalos, P.M., Du, D.Z., Graham, R.L. (eds.) Handbook of Combinatorial Optimization, pp. 455–531. Springer (2013)
- [13] Csirik, J., Galambos, G.: An  $O(n)$  bin-packing algorithm for uniformly distributed data. Computing 36(4), 313–319 (1986)
- [14] Csirik, J., Johnson, D.S., Kenyon, C.: On the worst-case performance of the sum-of-squares algorithm for bin packing. CoRR abs/cs/0509031 (2005)
- [15] Csirik, J., Johnson, D.S., Kenyon, C., Orlin, J.B., Shor, P.W., Weber, R.R.: On the sum-of-squares algorithm for bin packing. J. ACM 53, 1–65 (2006)

- [16] Csirik, J., Johnson, D.S., Kenyon, C., Shor, P.W., Weber, R.R.: A self organizing bin packing heuristic. In: Proc. 1st Meeting on Algorithm Engineering and Experiments (ALENEX). pp. 246–265. Springer-Verlag, London, UK, UK (1999)
- [17] Csirik, J., Woeginger, G.J.: Shelf algorithms for on-line strip packing. *Inform. Process. Lett.* 63, 171–175 (1997)
- [18] Degraeve, Z., Peeters, M.: Optimal integer solutions to industrial cutting-stock problems: Part 2, benchmark results. *INFORMS J. on Computing* 15(1), 58–81 (Jan 2003)
- [19] Floyd, S., Karp, R.M.: FFD bin packing for item sizes with uniform distributions on  $[0, 1/2]$ . *Algorithmica* 6(1-6), 222–240 (1991)
- [20] Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.* 3, 256–278 (1974)
- [21] Karp, R.M., Luby, M., Marchetti-Spaccamela, A.: Probabilistic analysis of multi-dimensional bin-packing problems. In: Proc. 16th Symp. on Theory of Computing (STOC). pp. 289–298 (1984)
- [22] Lee, C.C., Lee, D.T.: A simple online bin packing algorithm. *J. ACM* 32, 562–572 (1985)
- [23] Leighton, F.T., Shor, P.: Tight bounds for minimax grid matching with applications to the average case analysis of algorithms. *Combinatorica* 9, 161–187 (1989)
- [24] Murgolo, F.D.: Anomalous behavior in bin packing algorithms. *Discrete Appl. Math.* 21(3), 229–243 (1988)
- [25] Ramanan, P., Tsuga, K.: Average-case analysis of the modified harmonic algorithm. *Algorithmica* 4, 519–533 (1989)
- [26] Ramanan, P.V., Brown, D.J., Lee, C.C., Lee, D.T.: On-line bin packing in linear time. *J. Algorithms* 10, 305–326 (1989)
- [27] Rhee, W.T., Talagrand, M.: Exact bounds for the stochastic upward matching problem. *Trans. AMS* 307(1), 109–125 (1988)
- [28] Seiden, S.S.: On the online bin packing problem. *J. ACM* 49, 640–671 (2002)
- [29] Shor, P.W.: The average-case analysis of some online algorithms for bin packing. *Combinatorica* 6, 179–200 (1986)
- [30] Shor, P.W.: How to pack better than Best-Fit: Tight bounds for average-case on-line bin packing. In: Proc. 32nd Symp. on Foundations of Computer Science (FOCS). pp. 752–759 (1991)
- [31] Stille, W.M.: Solution Techniques for specific Bin Packing Problems with Applications to Assembly Line Optimization. Ph.D. thesis, TU Darmstadt (2008)
- [32] Xiaodong, G., Guoliang, C., Yinlong, X.: Deep performance analysis of refined harmonic bin packing algorithm. *J. Comput. Sci. Tech.* 17, 213–218 (2002)
- [33] Yao, A.C.C.: New algorithms for bin packing. *J. ACM* 27, 207–227 (1980)